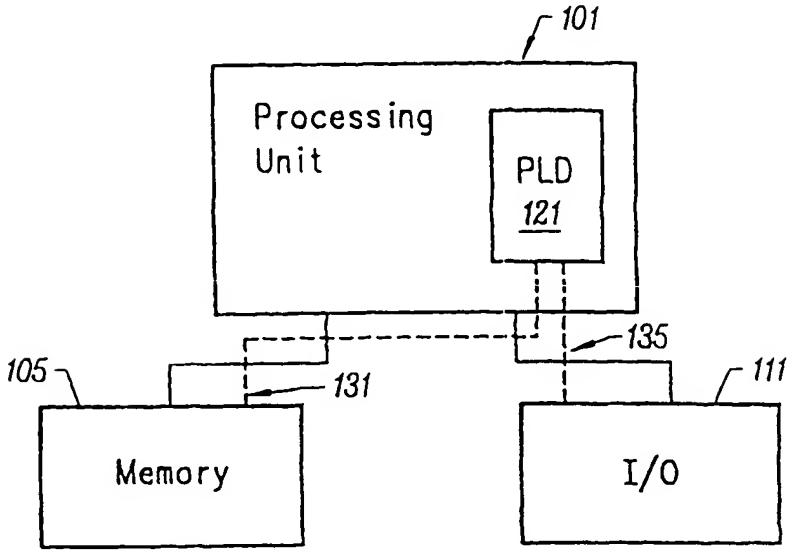
101
Processing Unit
PLD
121
135
105
131
111
Memory
I/O

FIG. 1

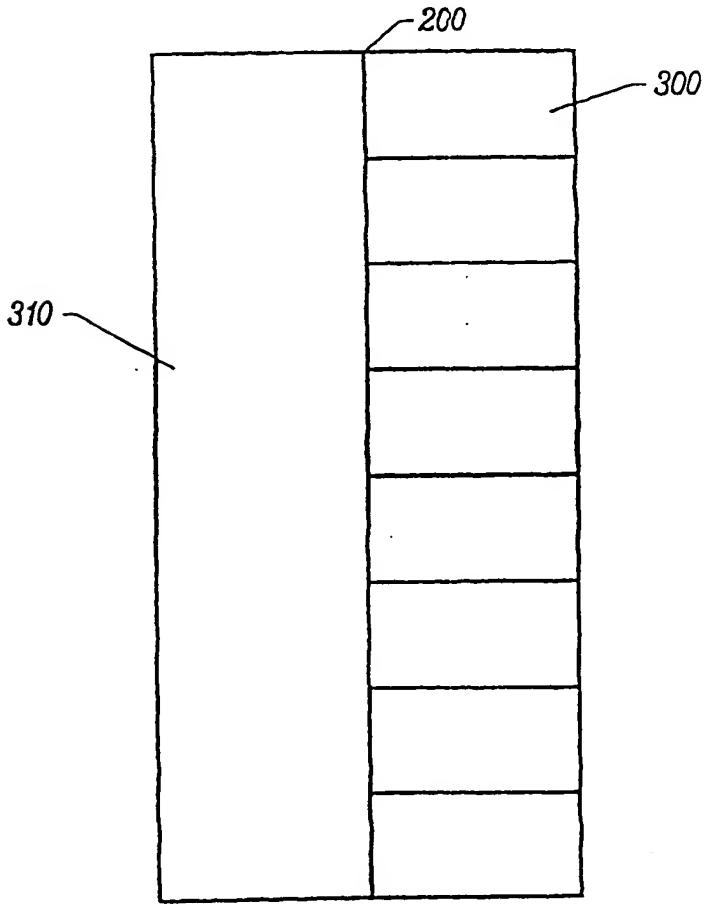
200
300
310

FIG. 3

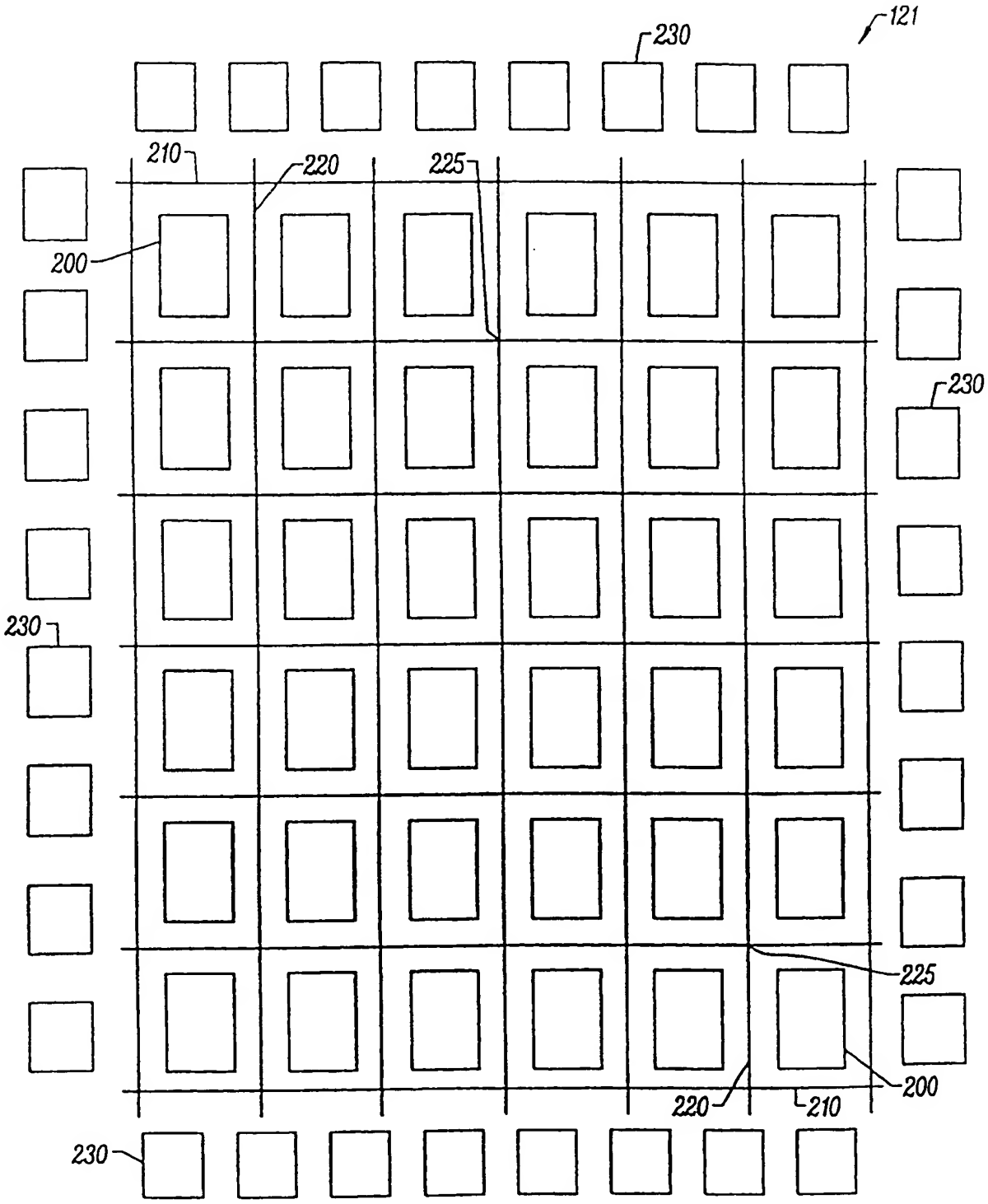

FIG. 2

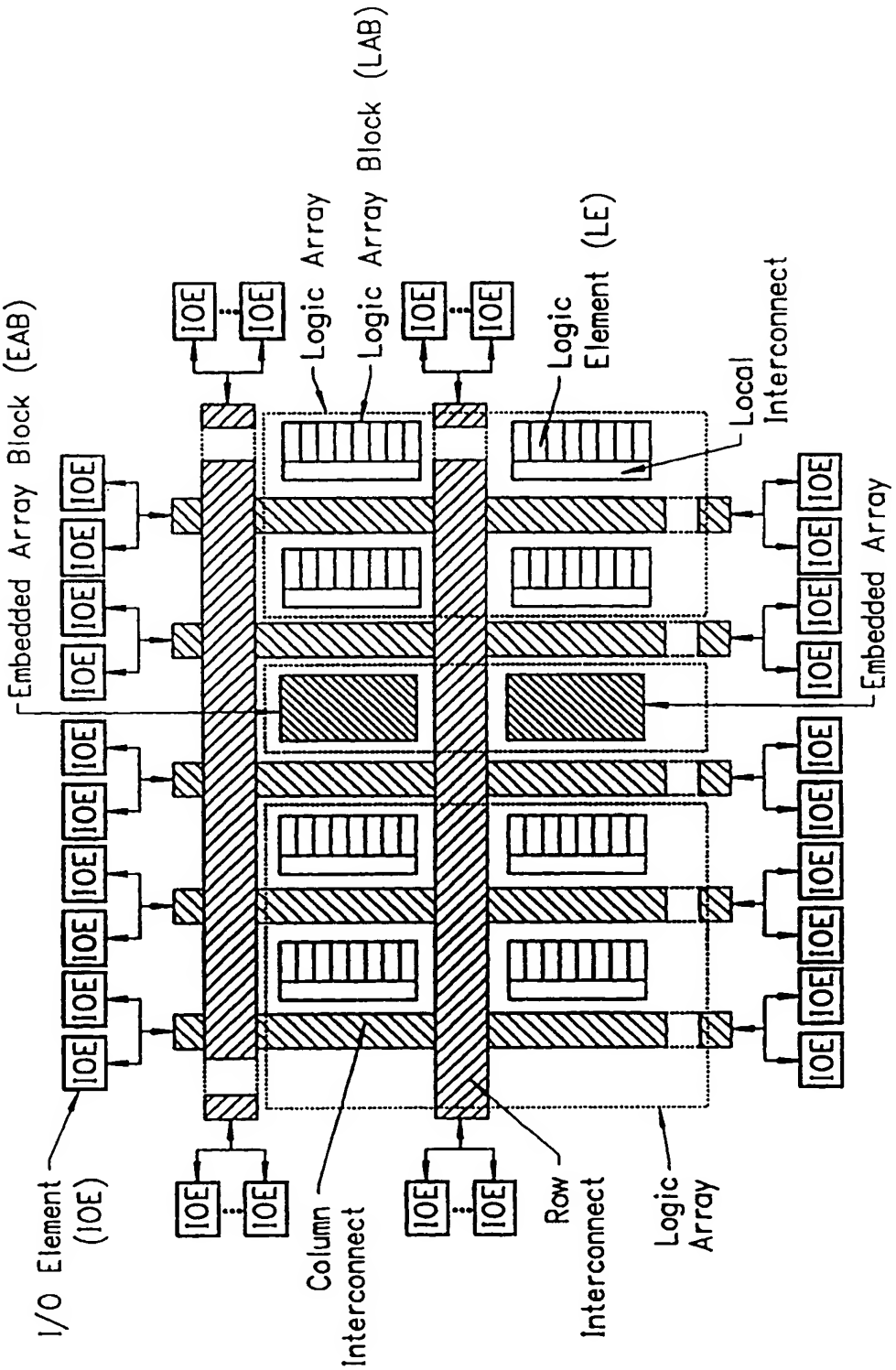
Embedded Array Block (EAB)
Logic Array
Logic Array Block (LAB)
Logic Element (LE)
Local Interconnect
Embedded Array
I/O Element (IOE)
Column Interconnect
Row Interconnect
Logic Array
IOE

FIG. 4

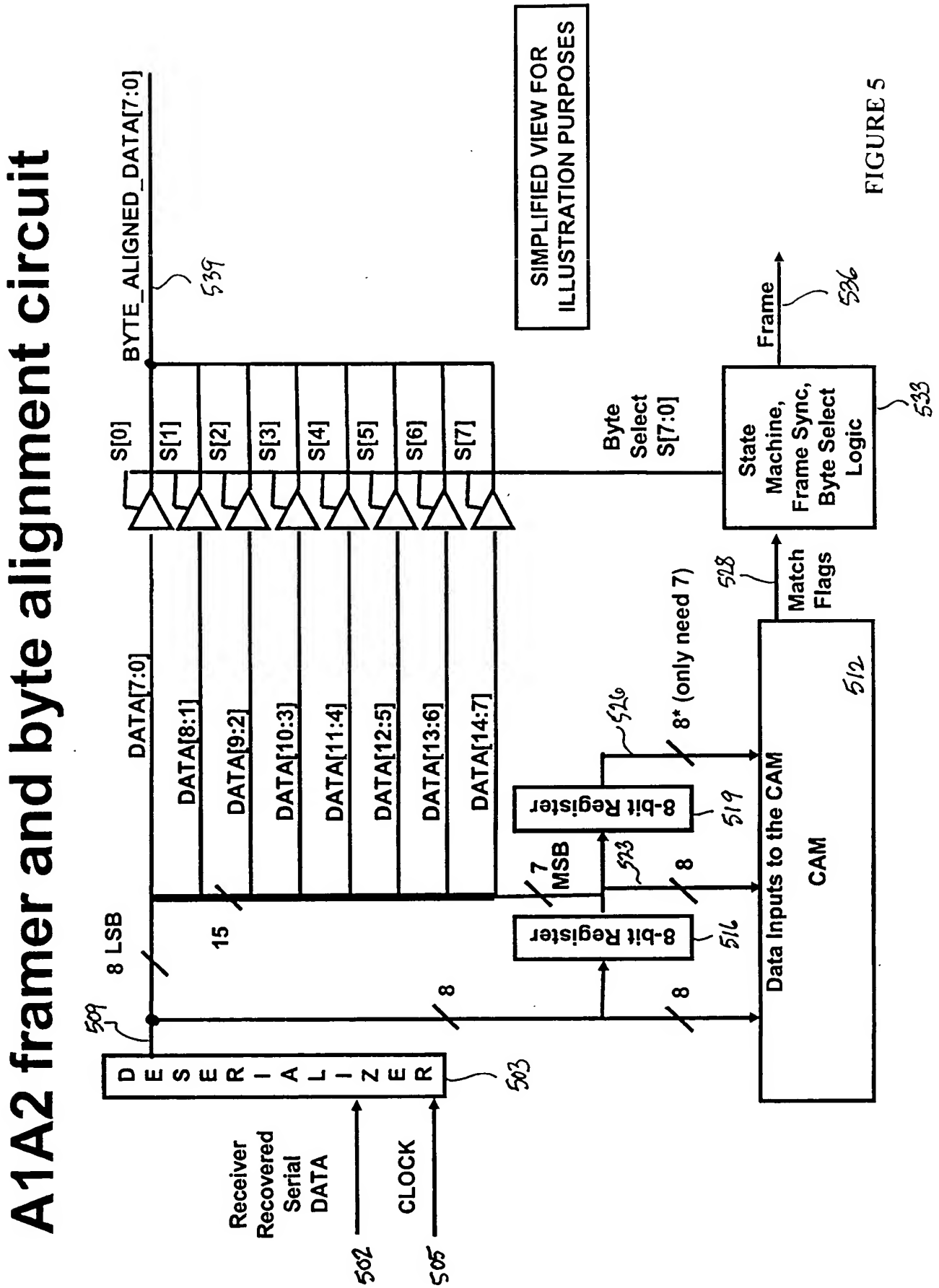
A1A2 framer and byte alignment circuit
SIMPLIFIED VIEW FOR ILLUSTRATION PURPOSES
BYTE_ALIGNED_DATA[7:0]
539
S[0]
S[1]
S[2]
S[3]
S[4]
S[5]
S[6]
S[7]
Byte Select S[7:0]
State Machine, Frame Sync, Byte Select Logic
533
Frame
536
528
Match Flags
DATA[7:0]
DATA[8:1]
DATA[9:2]
DATA[10:3]
DATA[11:4]
DATA[12:5]
DATA[13:6]
DATA[14:7]
8* (only need 7)
526
8-bit Register
519
8-bit Register
516
7 MSB
523
8
8
8
15
8 LSB
509
Data Inputs to the CAM
CAM
512
DESERIALIZER
503
Receiver Recovered Serial DATA
CLOCK
502
505

FIGURE 5

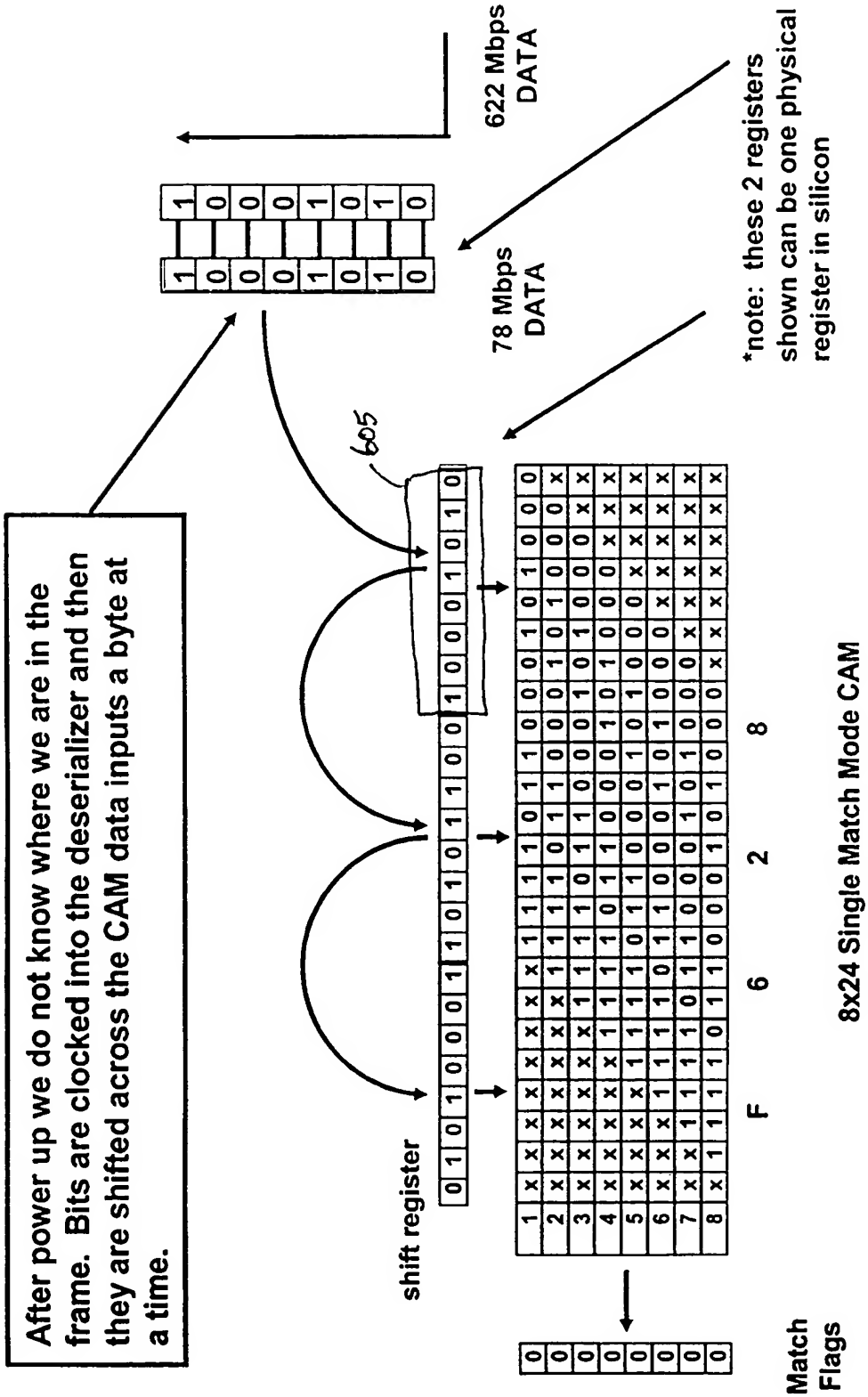
A1A2 Framer using CAM
After power up we do not know where we are in the frame. Bits are clocked into the deserializer and then they are shifted across the CAM data inputs a byte at a time.
622 Mbps DATA
78 Mbps DATA
*note: these 2 registers shown can be one physical register in silicon
605
shift register
F 6 2 8
8x24 Single Match Mode CAM
Match Flags

FIGURE 6

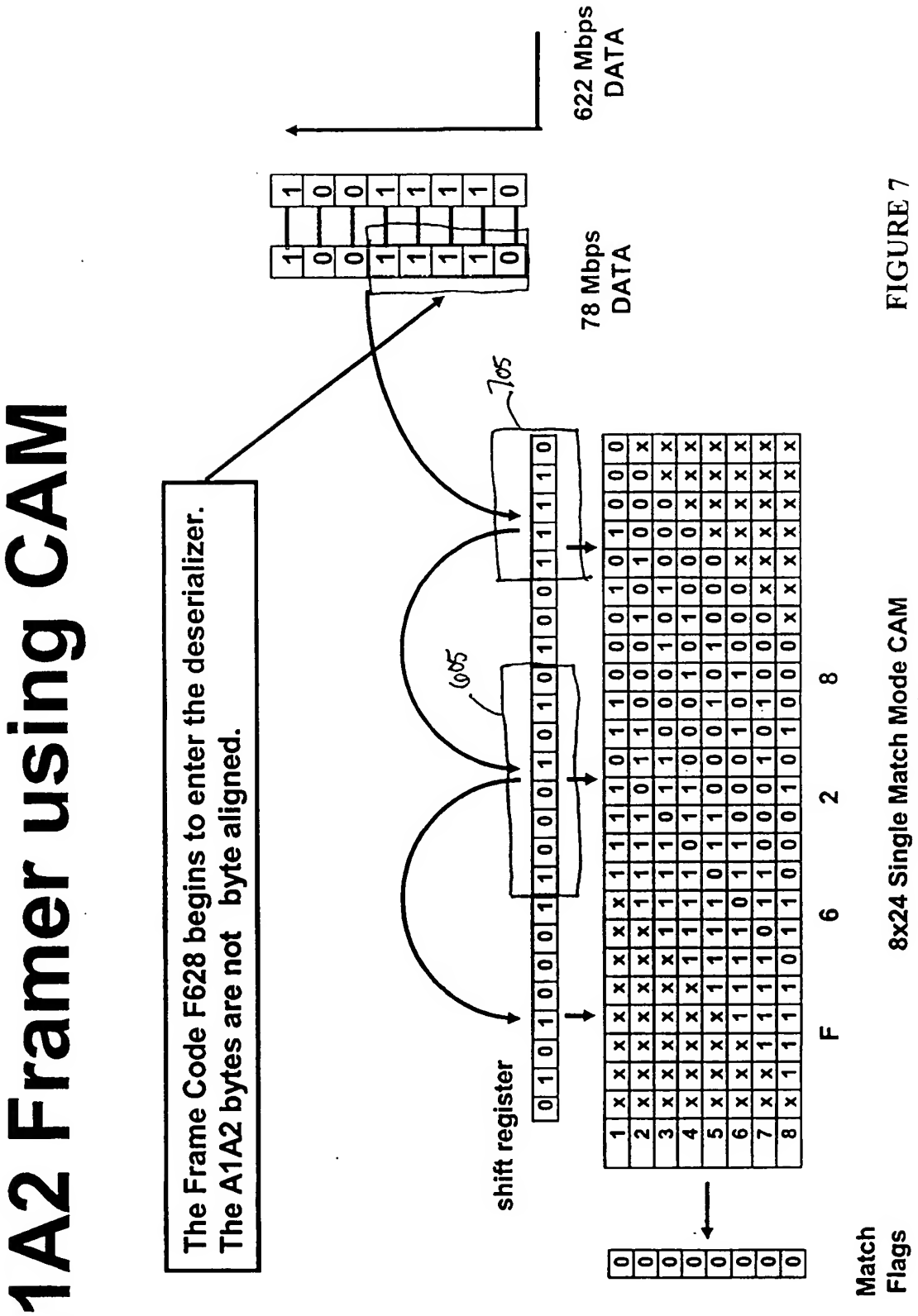
A1A2 Framer using CAM
The Frame Code F628 begins to enter the deserializer.
The A1A2 bytes are not byte aligned.
622 Mbps DATA
78 Mbps DATA
shift register
105
105
F 6 2 8
8x24 Single Match Mode CAM
(can be 8x23)
Match Flags

FIGURE 7

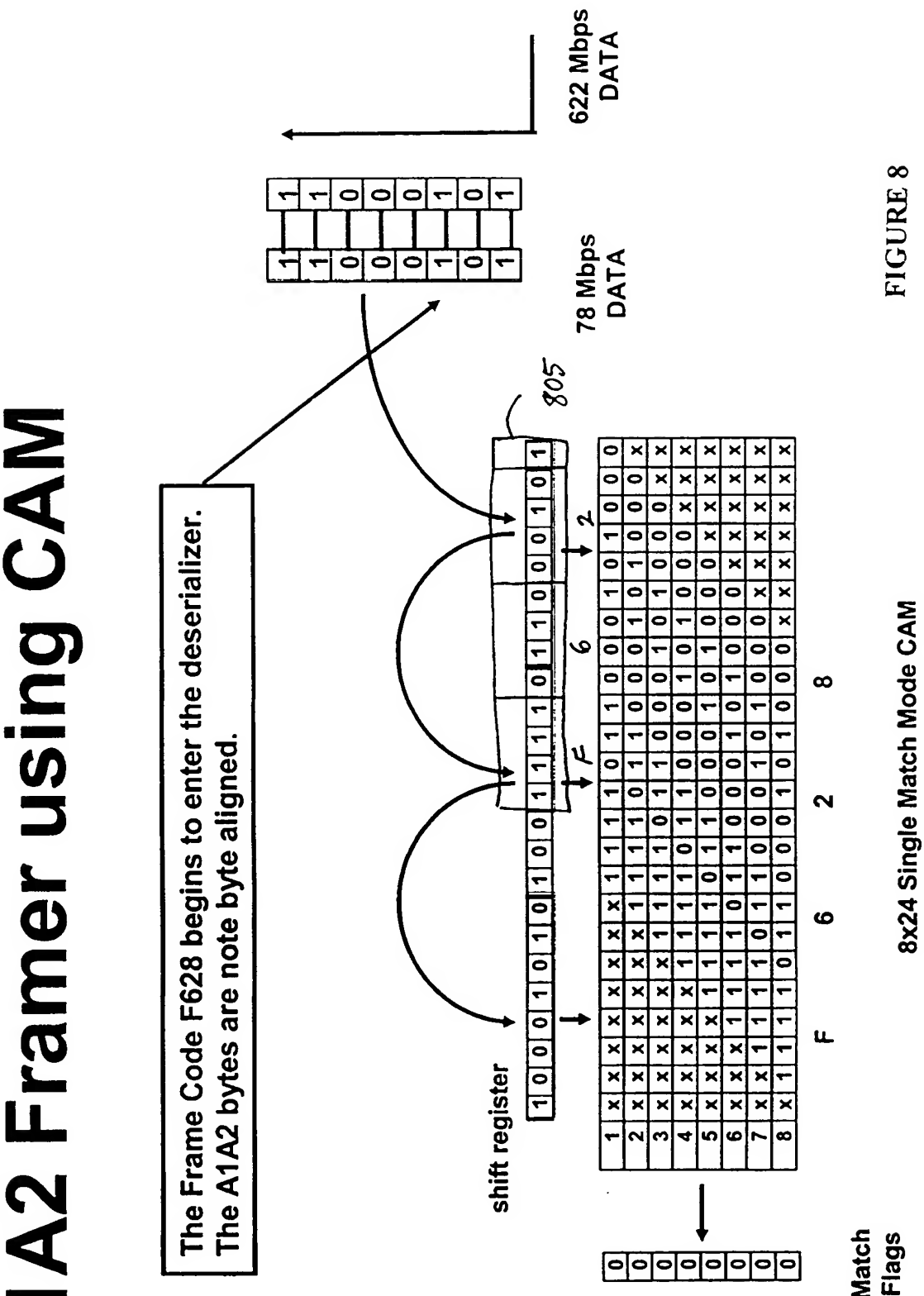
A1A2 Framer using CAM
The Frame Code F628 begins to enter the deserializer.
The A1A2 bytes are note byte aligned.
622 Mbps DATA
78 Mbps DATA
805
shift register
F
6
2
8
Match Flags
8x24 Single Match Mode CAM

FIGURE 8

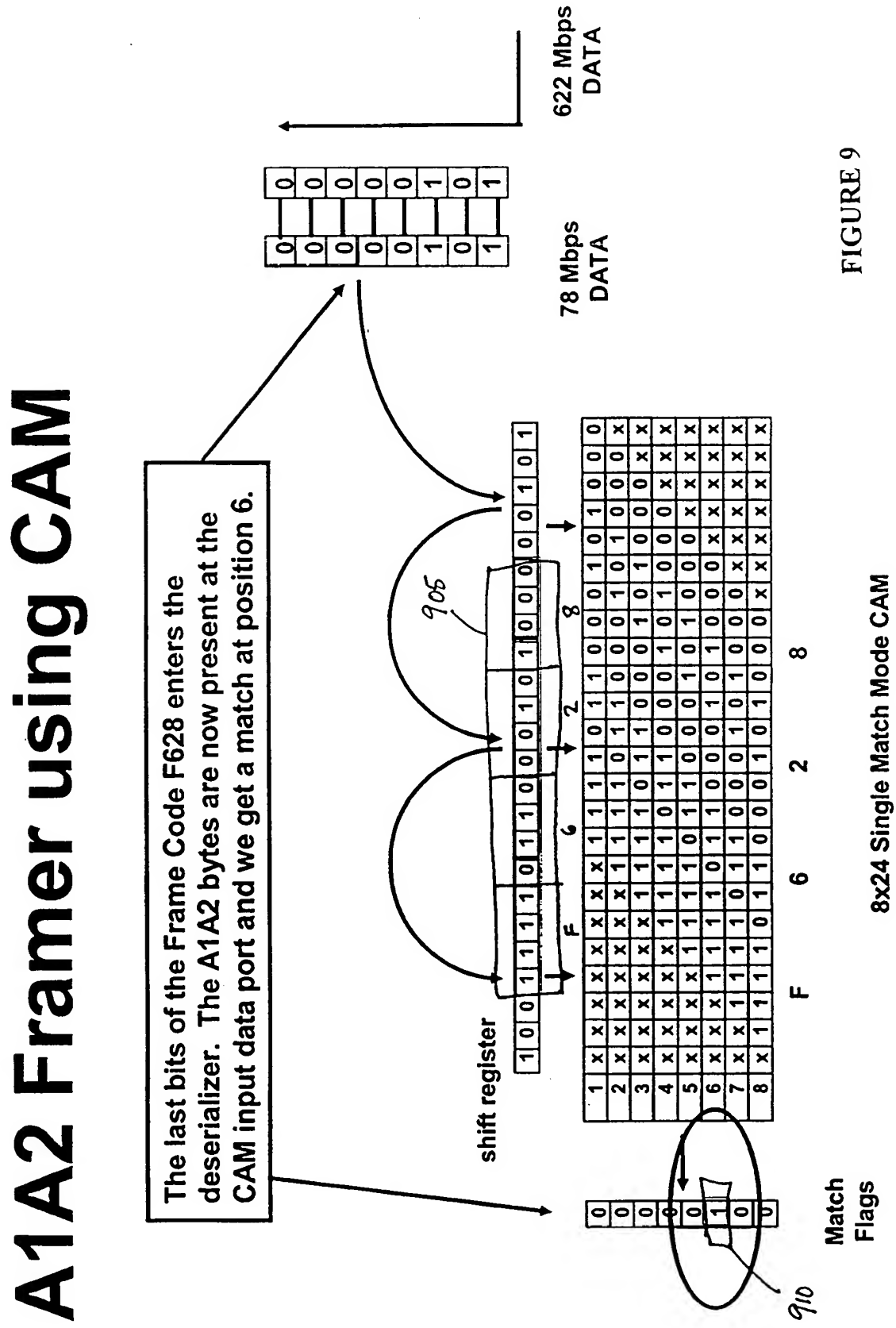
A1A2 Framer using CAM
The last bits of the Frame Code F628 enters the deserializer. The A1A2 bytes are now present at the CAM input data port and we get a match at position 6.
622 Mbps DATA
78 Mbps DATA
shift register
905
910
Match Flags
F 6 2 8
8x24 Single Match Mode CAM

FIGURE 9

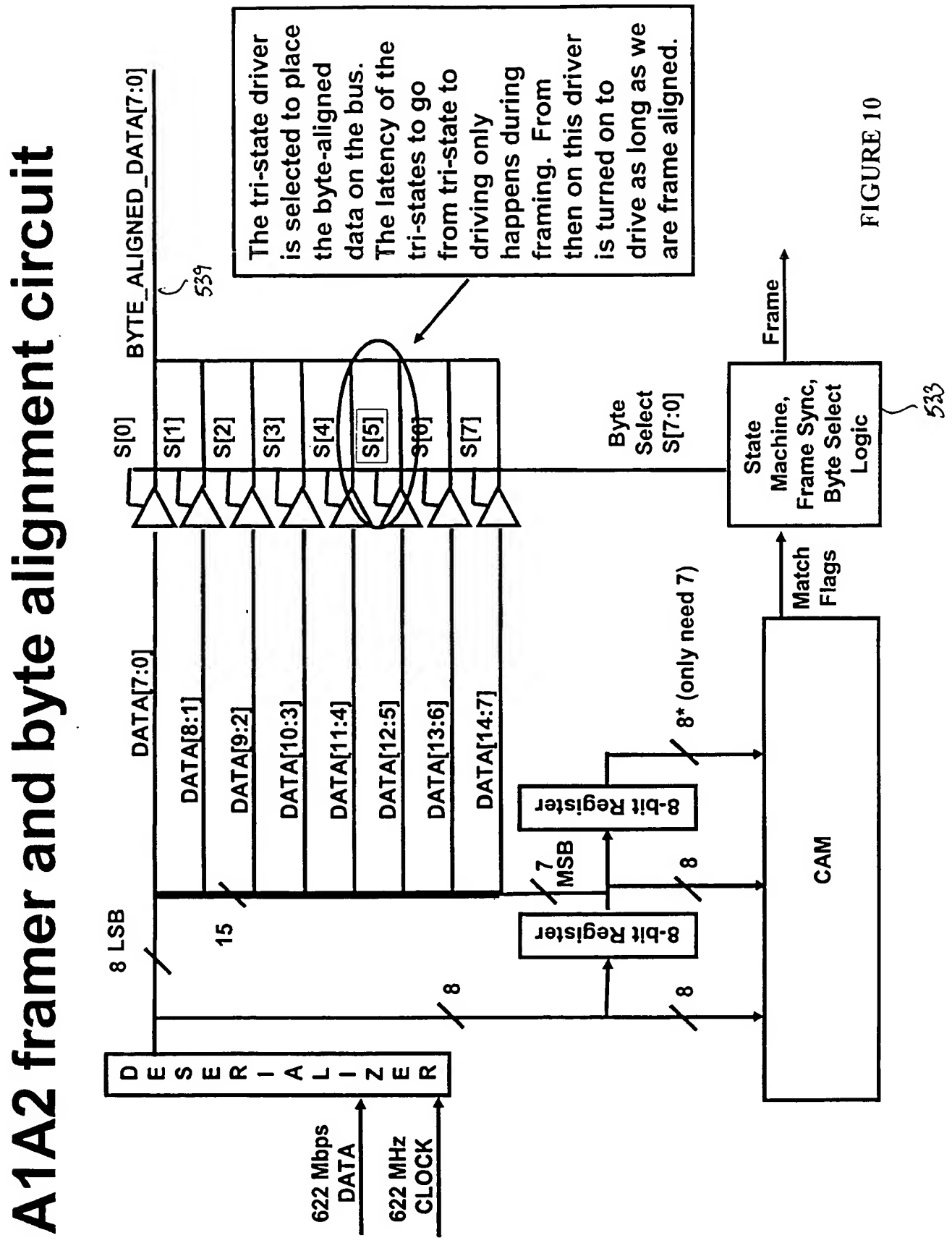
A1A2 framer and byte alignment circuit
The tri-state driver is selected to place the byte-aligned data on the bus. The latency of the tri-states to go from tri-state to driving only happens during framing. From then on this driver is turned on to drive as long as we are frame aligned.
BYTE_ALIGNED_DATA[7:0]
539
S[0]
S[1]
S[2]
S[3]
S[4]
S[5]
S[6]
S[7]
Byte Select S[7:0]
State Machine, Frame Sync, Byte Select Logic
533
Frame
Match Flags
CAM
8* (only need 7)
8-bit Register
8-bit Register
7 MSB
8
8
8
15
8 LSB
DATA[7:0]
DATA[8:1]
DATA[9:2]
DATA[10:3]
DATA[11:4]
DATA[12:5]
DATA[13:6]
DATA[14:7]
DESERIALIZER
622 Mbps DATA
622 MHz CLOCK

FIGURE 10

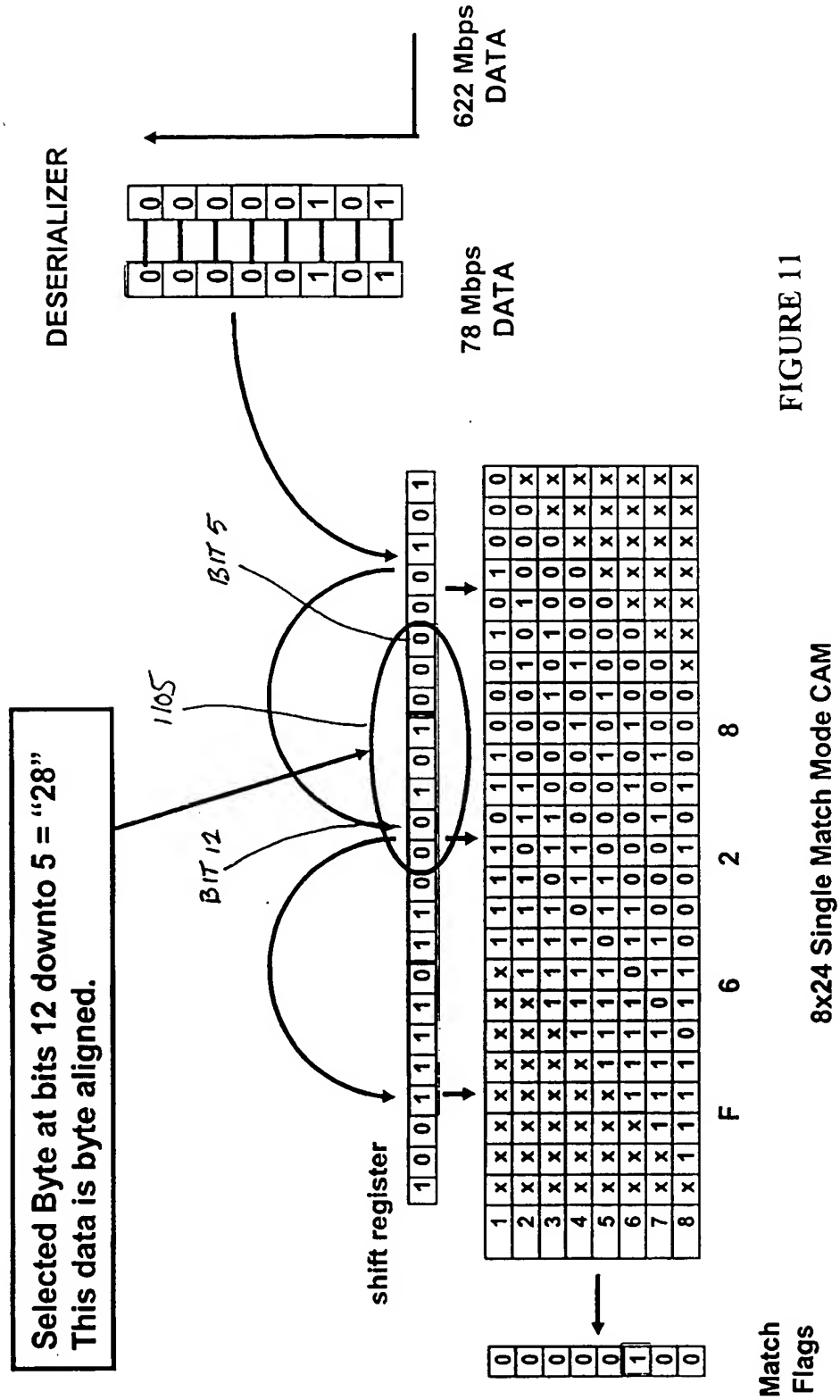
A1A2 Framer using CAM
DESERIALIZER
622 Mbps DATA
78 Mbps DATA
Selected Byte at bits 12 downto 5 = "28"
This data is byte aligned.
BIT 5
BIT 12
105
shift register
F 6 2 8
8x24 Single Match Mode CAM
Match Flags

FIGURE 11

Extending the idea to 20-bit bus

- Serial to Parallel bus can be 8-bit, 10-bit, 16-bit or 20-bit.
- We will need to be able to operate with any one of these bus configurations.
- This shows how data would map from the serial to parallel converter to the shift register and into the CAM for a 20-bit parallel data SERDES.

Need SERDES parallel register (20) + pattern length A1A2 (16) - 1 = total number of shift register bits wide

Here we have a 35-bit shift register

The CAM depth is the number of SERDES parallel bits = 20

So, the CAM would need to be 20 x 35

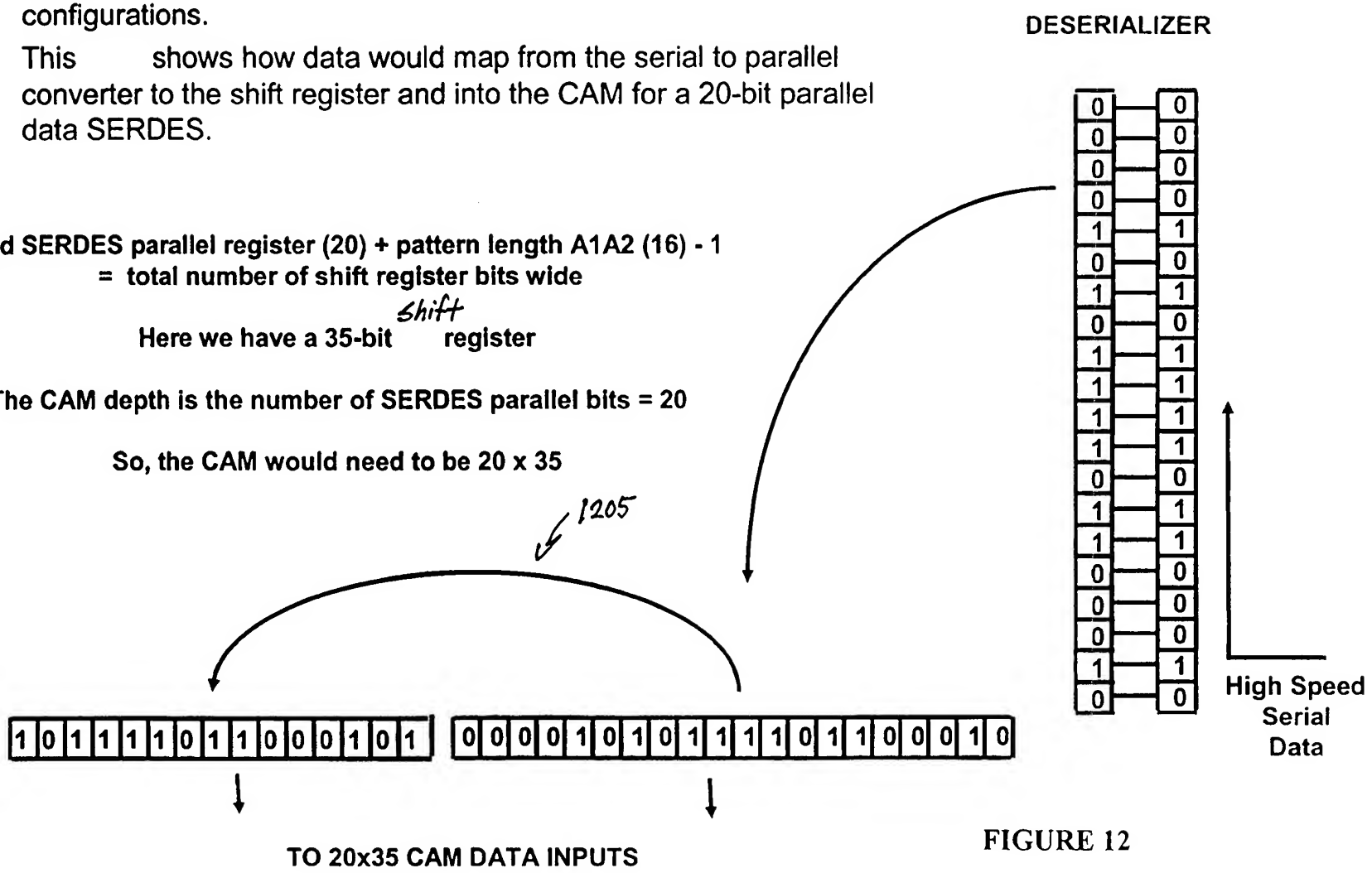

FIGURE 12

Application No.: 015114-064700US
Applicant: Amanda Noe
Title: PATTERN DETECT AND BYTE ALIGN...
Sheet 11 of 26

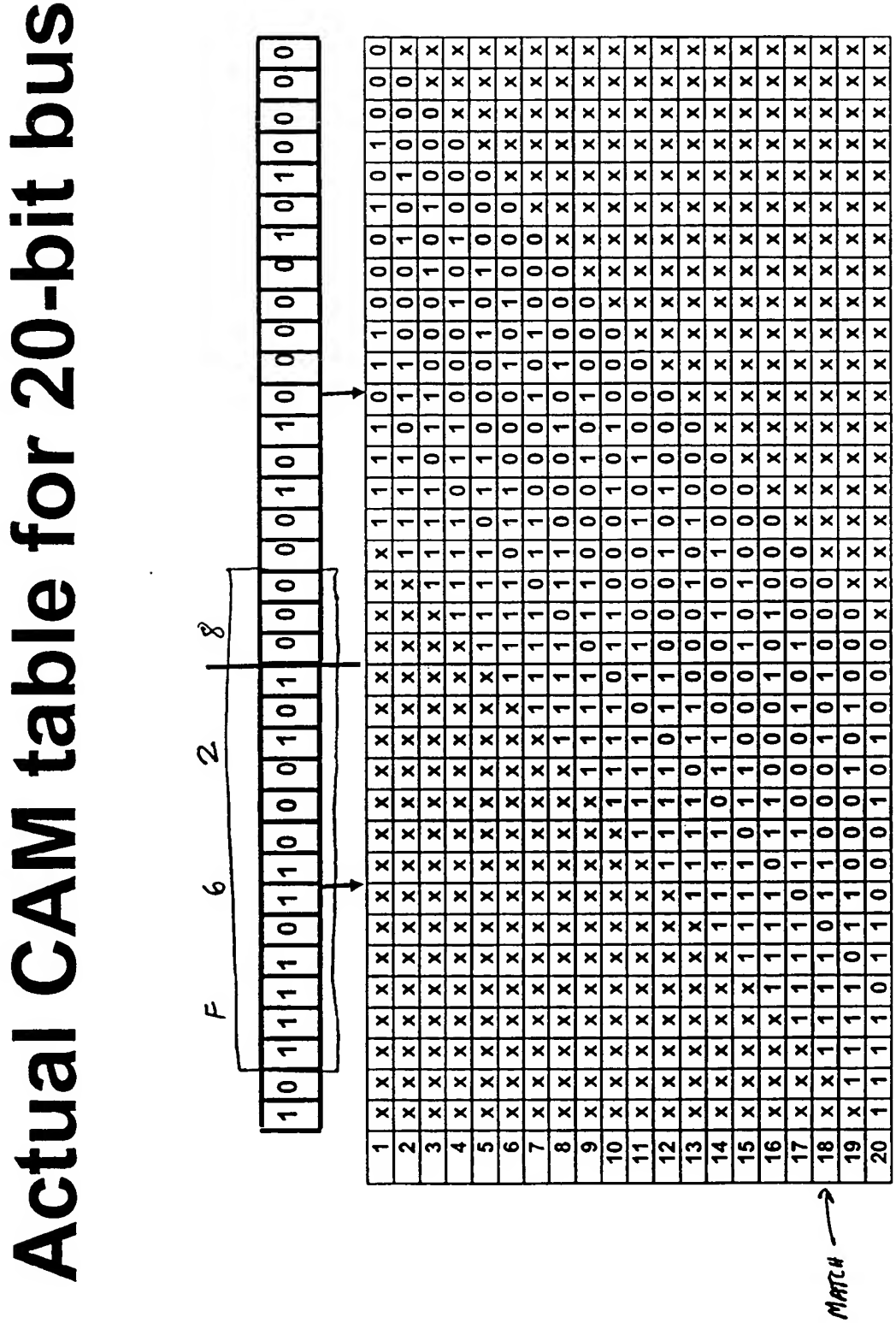

FIGURE 13

Extending idea to 16-bit bus

- This shows how data would map from the serial to parallel converter to the shift register and into the CAM for a 16-bit parallel data SERDES.

Need SERDES parallel register (16) + pattern length A1A2 (16) - 1
total number of shift register bits wide = 31

Here we have a 35-bit shift register and map the bits the same way as before
We will just set the 4 bits that are not used to '0'

The CAM depth is the number of SERDES parallel bits = 16

So, the CAM would need to be 16 x 35, although it will be 20x35
because the 20-bit case set the depth requirement
Entries 17 to 20 will be all U'

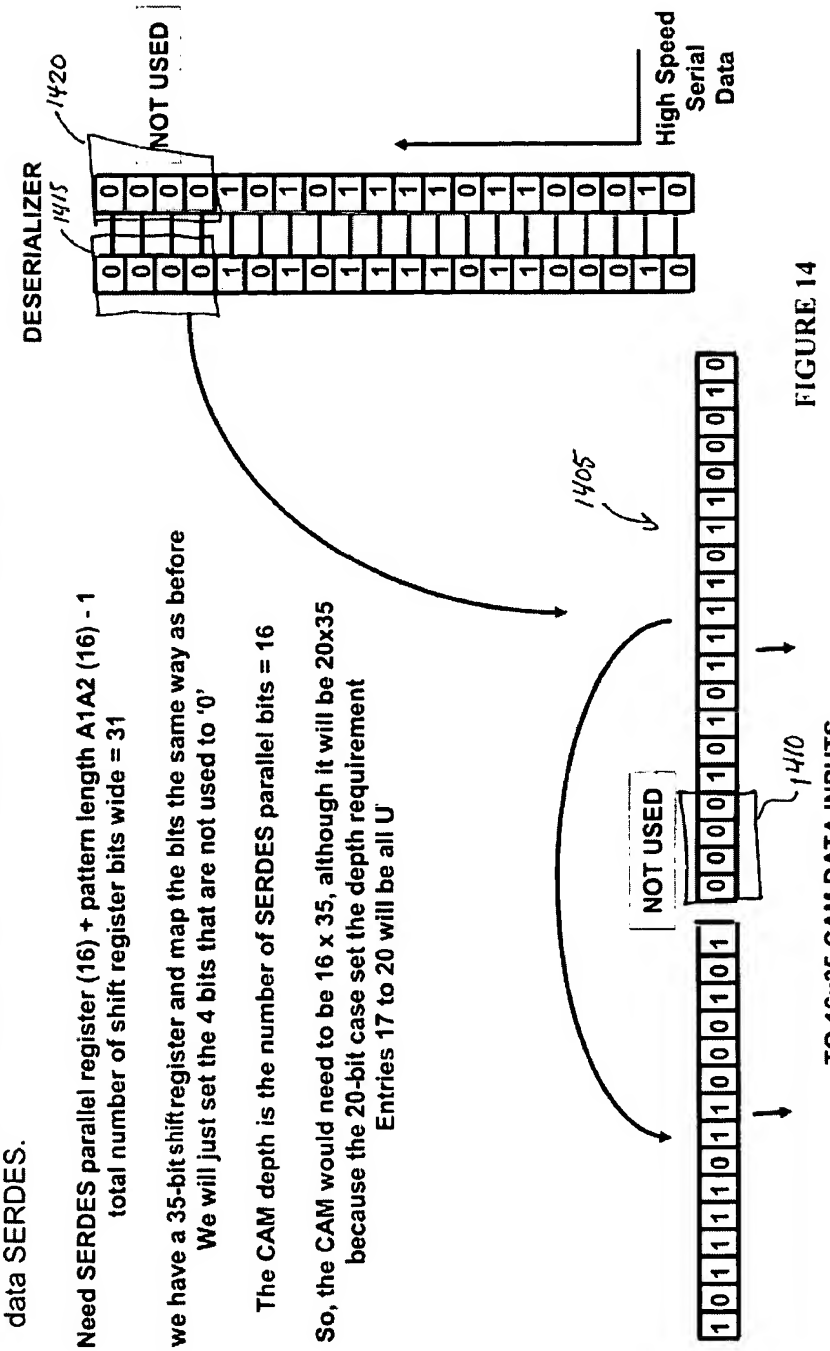

FIGURE 14

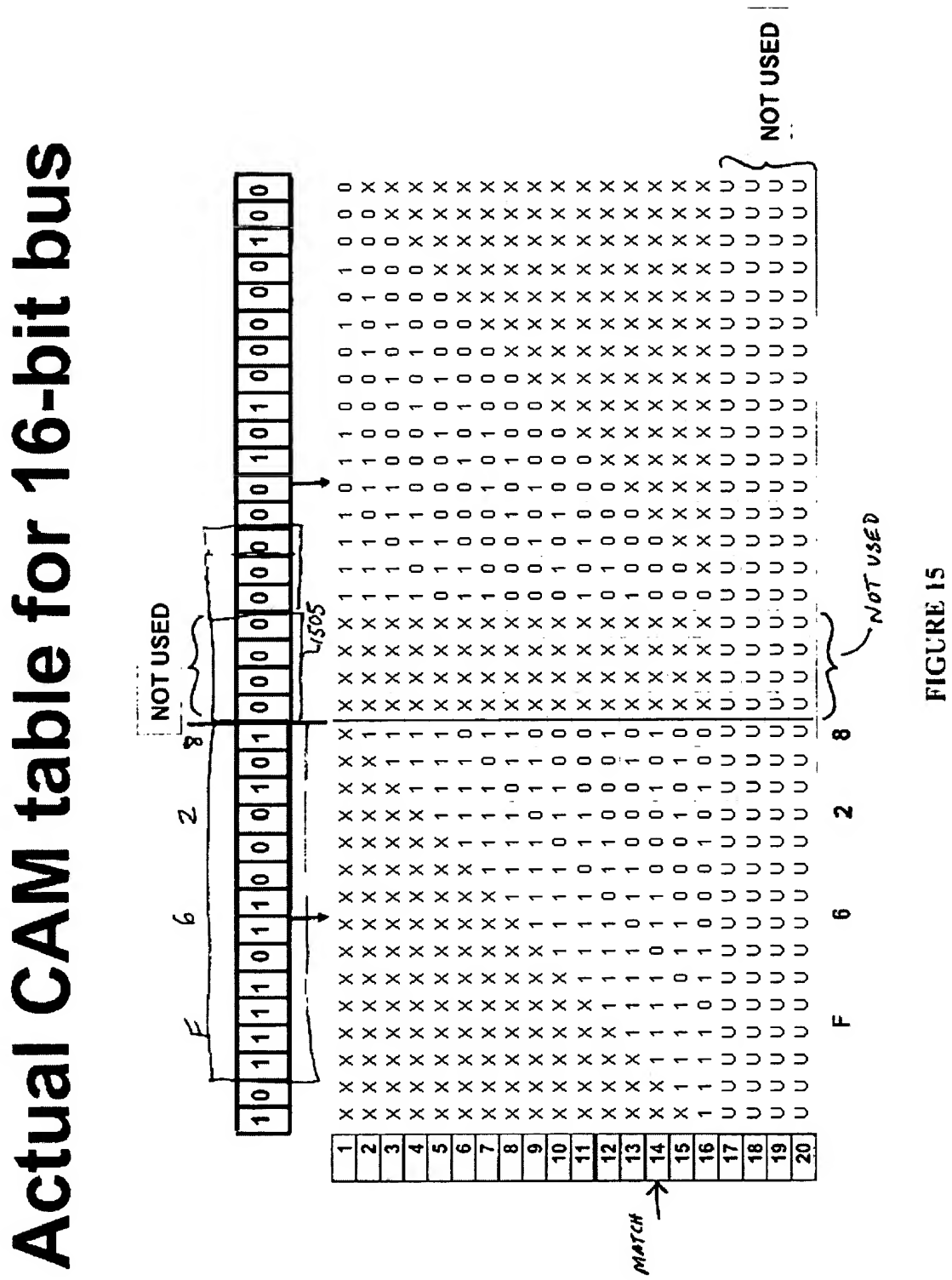
Actual CAM table for 16-bit bus
NOT USED
NOT USED
NOT USED
MATCH
F
6
2
8
E
6
2
8

FIGURE 15

Extending idea to 10-bit bus

- This slide shows how data would map from the serial to parallel converter to the shift register and into the CAM for a 10-bit parallel data SERDES.

We will assume that the CAM size has already been set by the 20-bit bus width case and so we will use muxes as shown below for the 10-bit bus width case. Alternatively, we could have increased the size of the CAM to eliminate the need for the muxes.

Need SERDES parallel register (10) + pattern length A1A2 (16) - 1
total number of shift register bits wide = 25

Here we have a 35-bit shift register and map the bits a little differently
We set the 10 bits that are not used to '0', use muxes to shift the data
The first 20-bit shift register is now 2 virtual 10-bit shift registers
The CAM depth is the number of SERDES parallel bits = 10

So, the CAM would need to be 10 x 35, although it will be 20x35
because the 20-bit case set the depth requirement
Entries 11 to 20 will be all U

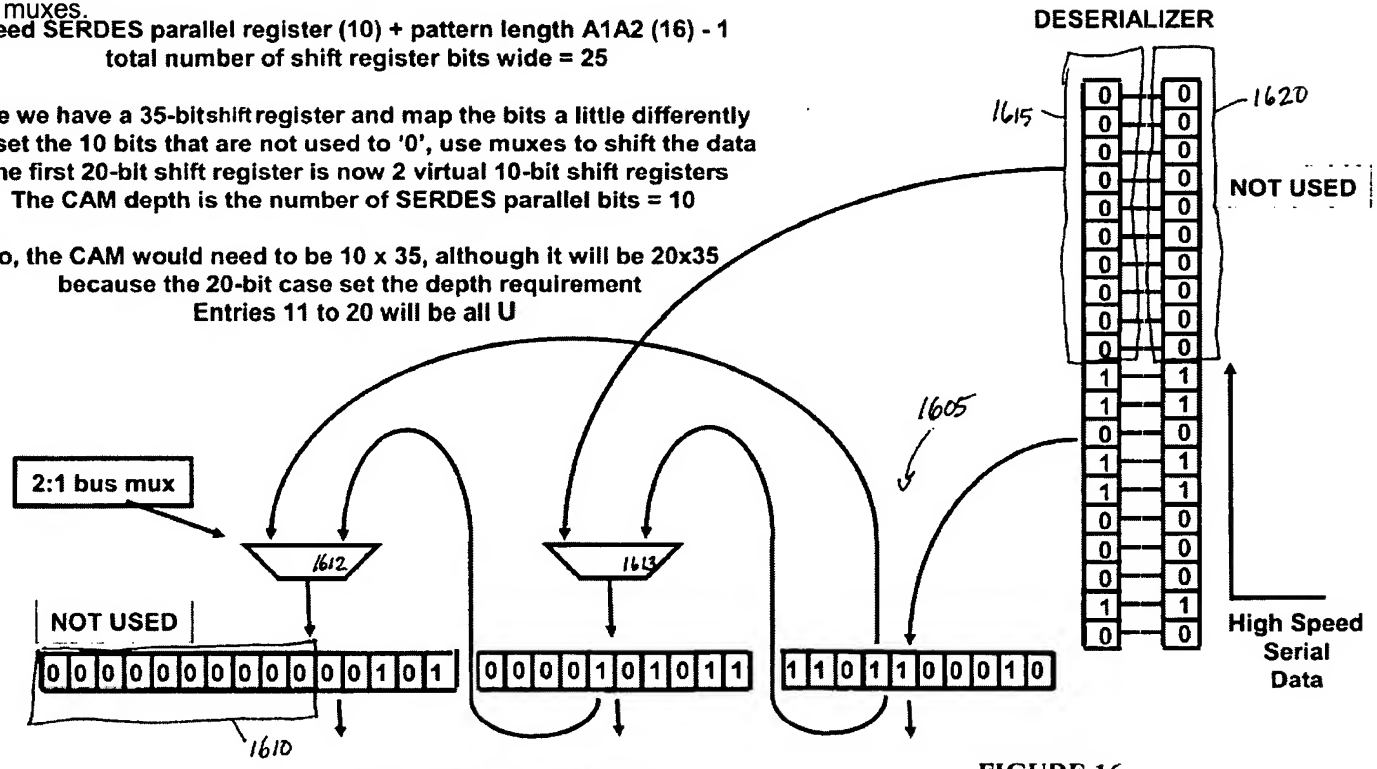

FIGURE 16

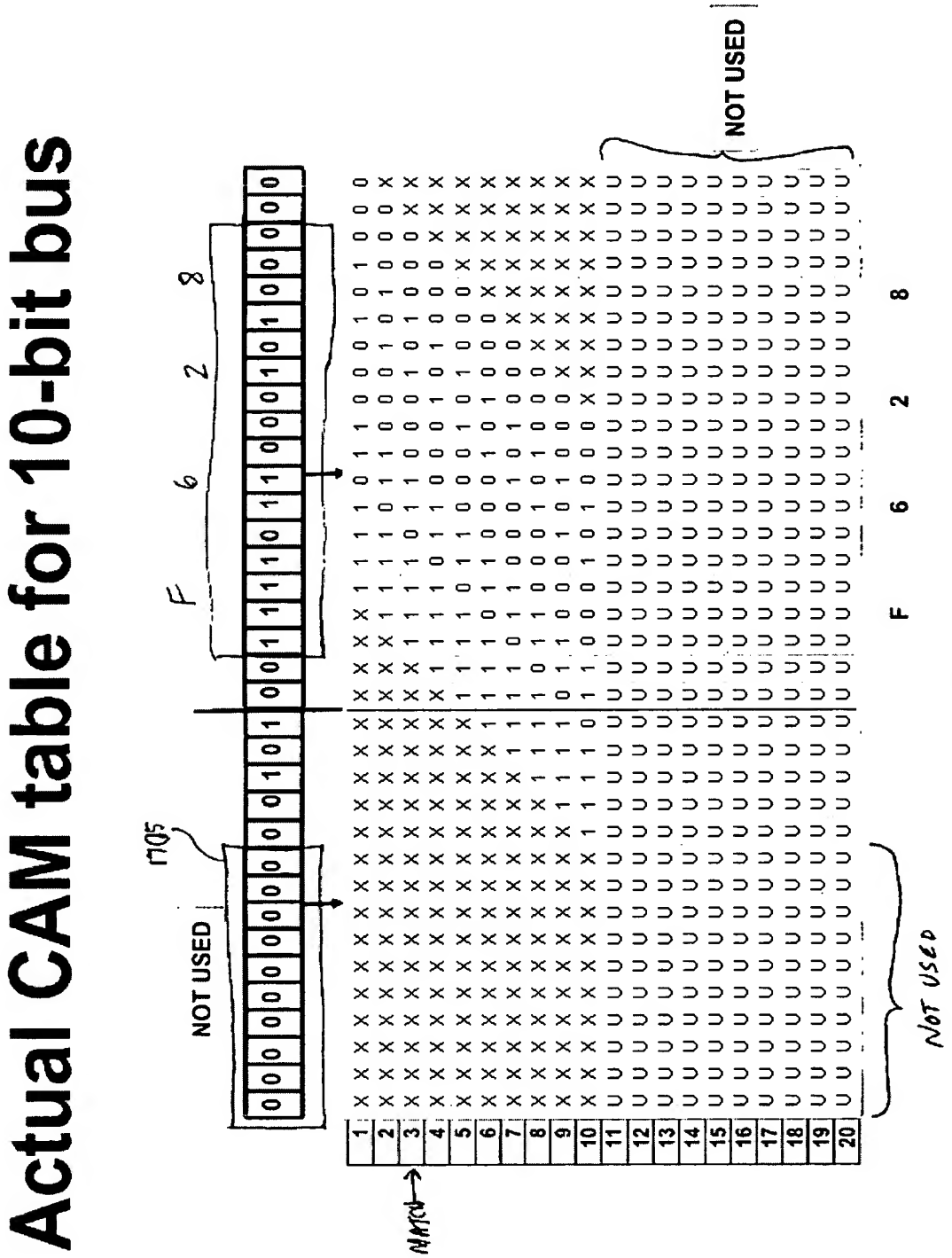
Actual CAM table for 10-bit bus
NOT USED
F
6
2
8
NOT USED
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

FIGURE 17

Extending idea to 8-bit bus

- This slide shows how data would map from the serial to parallel converter to the shift register and into the CAM for a 8-bit parallel data SERDES.

Need SERDES parallel register (8) + pattern length A1A2 (16) - 1
total number of shift register bits wide = 23
The actual width is 27 due to (2) 2-bit gaps
H re we have a 35-bitshift register and map the bits a little differently
We set the 10 bits that are not used to '0', use muxes to shift the data
The first 20-bit shift register is now 2 virtual 10-bit shift registers
The CAM depth is the number of SERDES parallel bits = 8
We have 2 gaps where the 10-bit data bits 9 and 10 are skipped over
S , the CAM would need to be 8 x 35, although it will be 20x35
because the 20-bit case set the depth requirement
Entries 9 to 20 will be all U

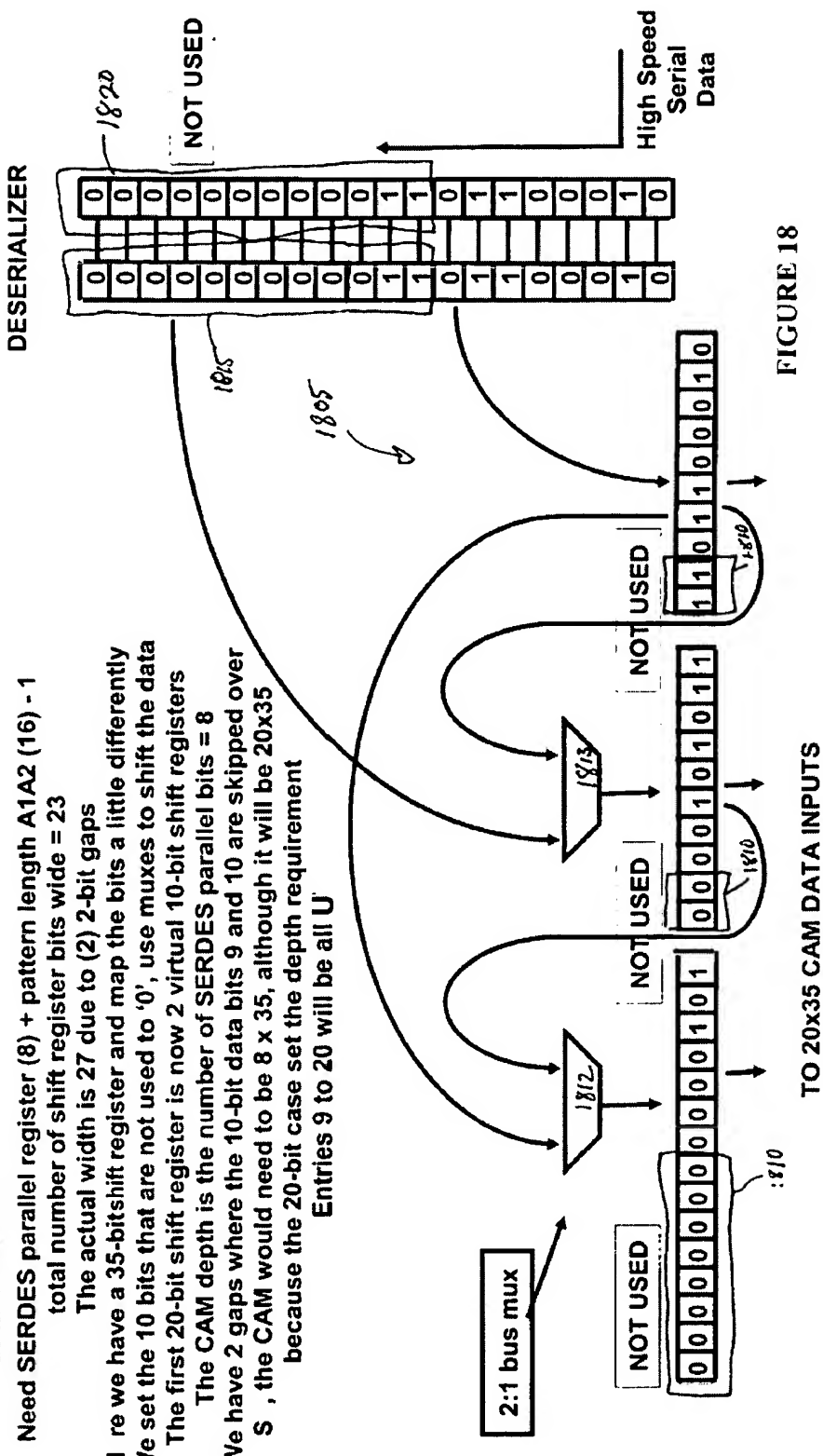

FIGURE 18

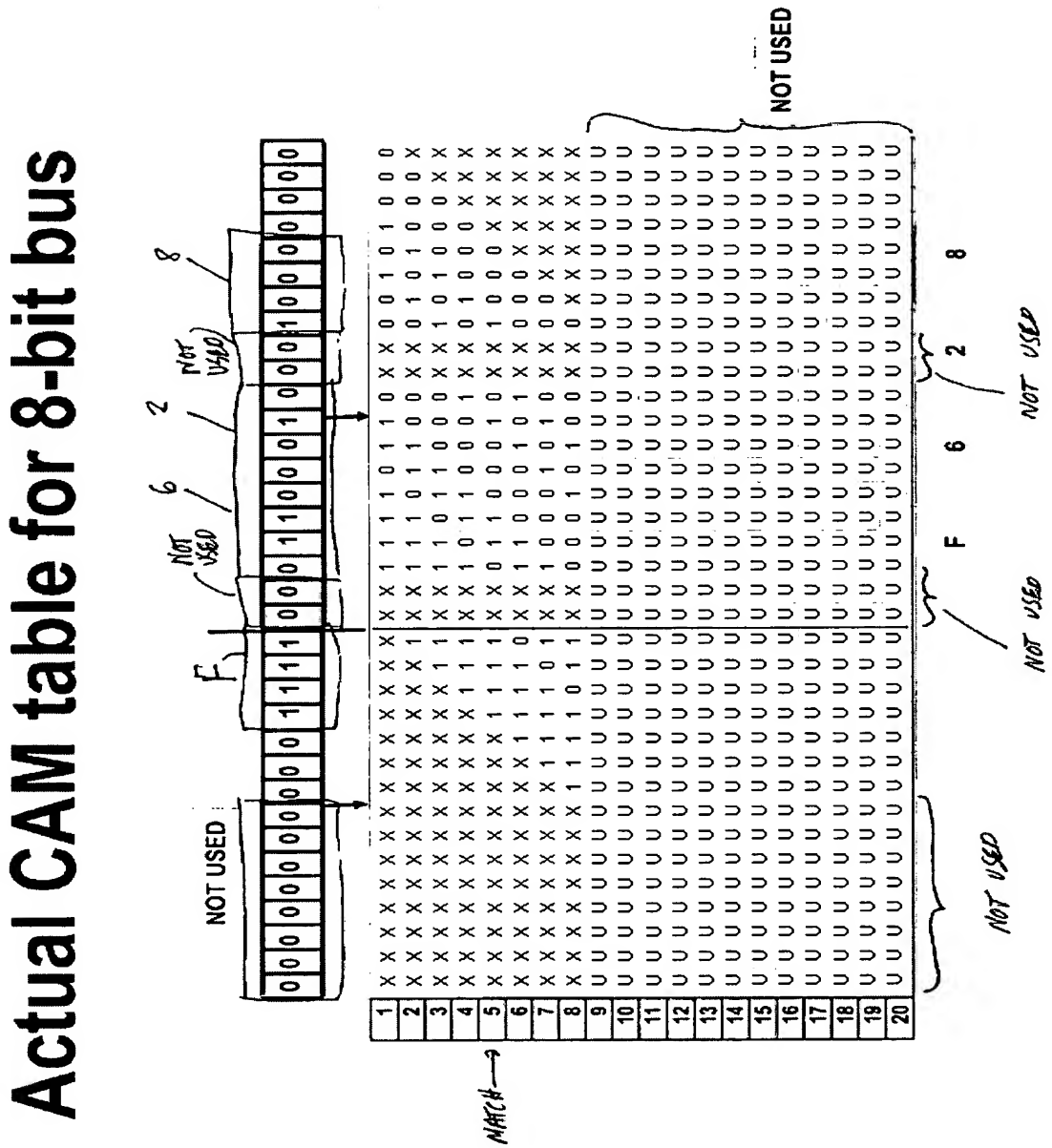
Actual CAM table for 8-bit bus
NOT USED
NOT USED
NOT USED
NOT USED
NOT USED
NOT VALID
MATCH
F
6
2
8

FIGURE 19

CAM Table for matching multiple F6's followed by multiple 28's for a 20-bit bus width

When matching a SONET framing pattern F6F62828, the state machine would look for two consecutive matches in the top half of the table followed by two corresponding consecutive matches in the bottom half of the table.

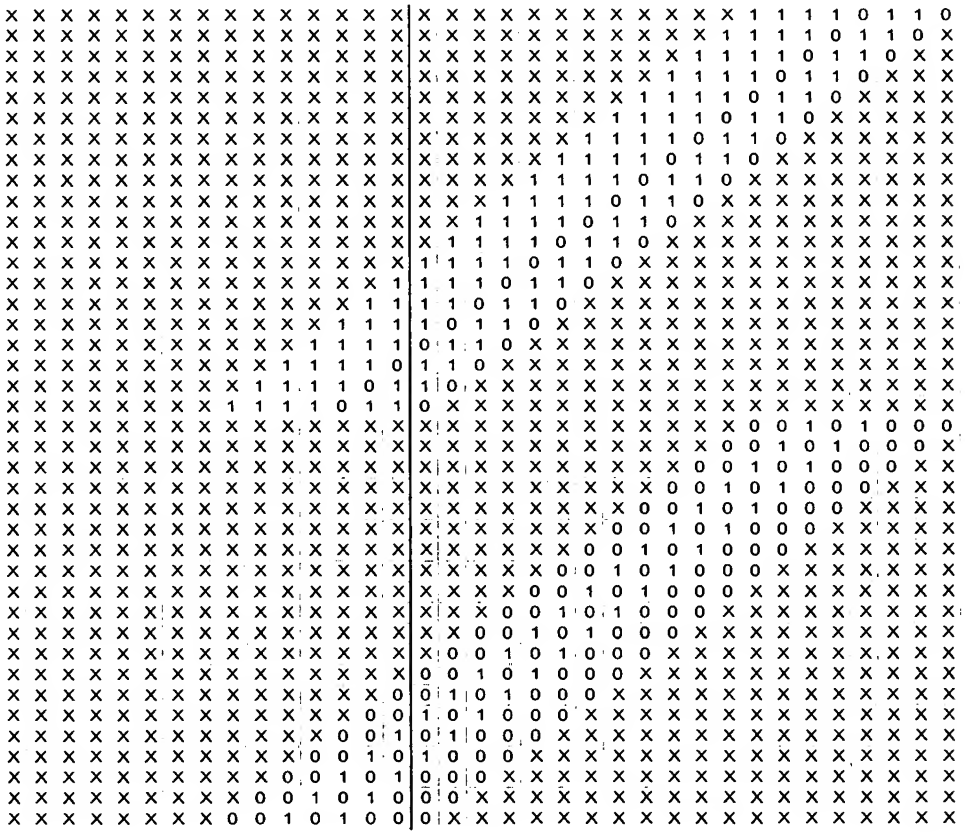

Figure 20

CAM Table for matching multiple F6's followed by multiple 28's for a 16-bit bus width

When matching a SONET framing pattern F6F62828, the state machine would look for two consecutive matches in the top half of the table followed by two corresponding consecutive matches in the bottom half of the table.

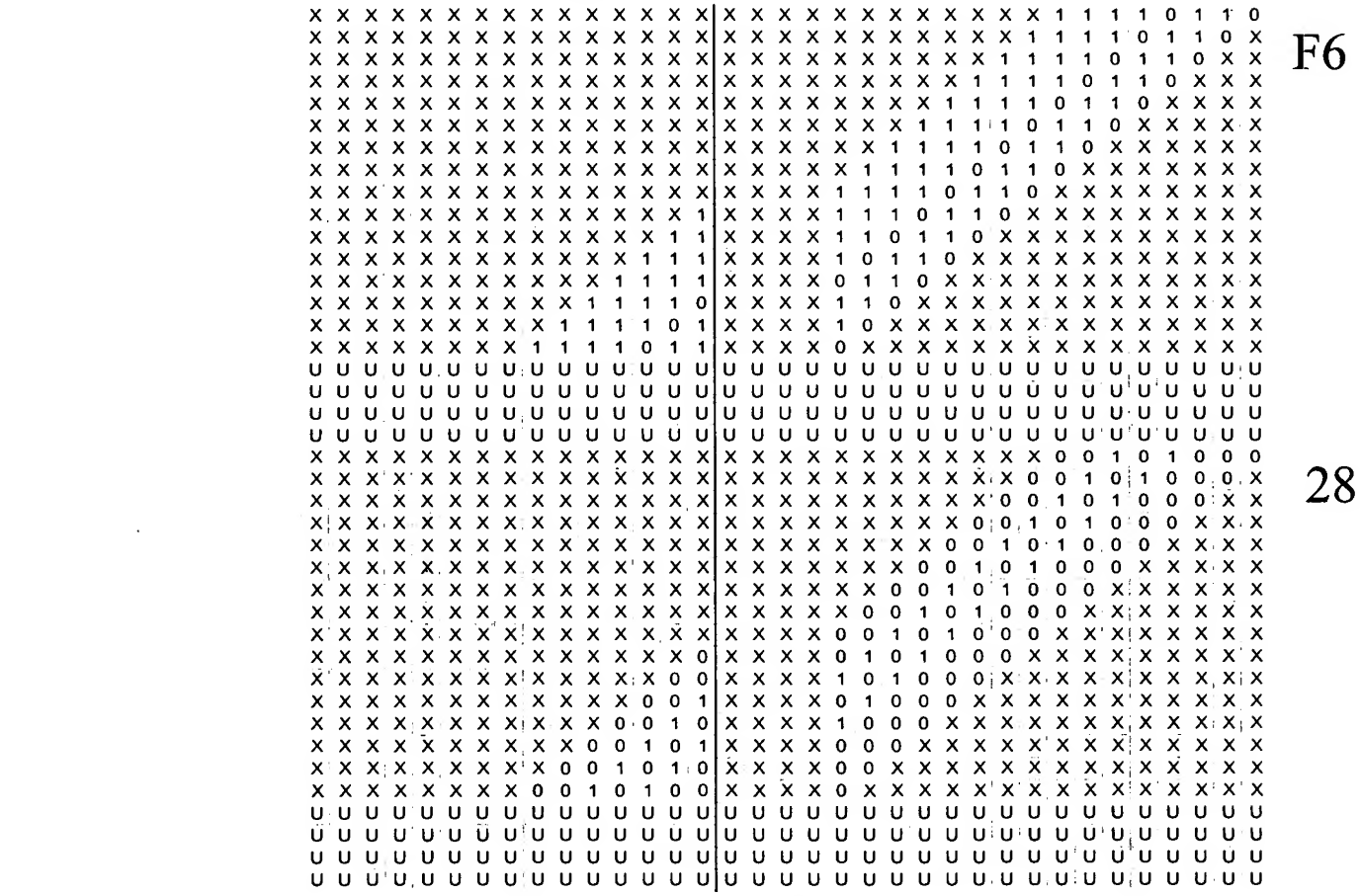

Figure 21

CAM Table for matching multiple F6's followed by multiple 28's for a 10-bit bus width

When matching a SONET framing pattern F6F62828, the state machine would look for two consecutive matches in the top half of the table followed by two corresponding consecutive matches in the bottom half of the table.

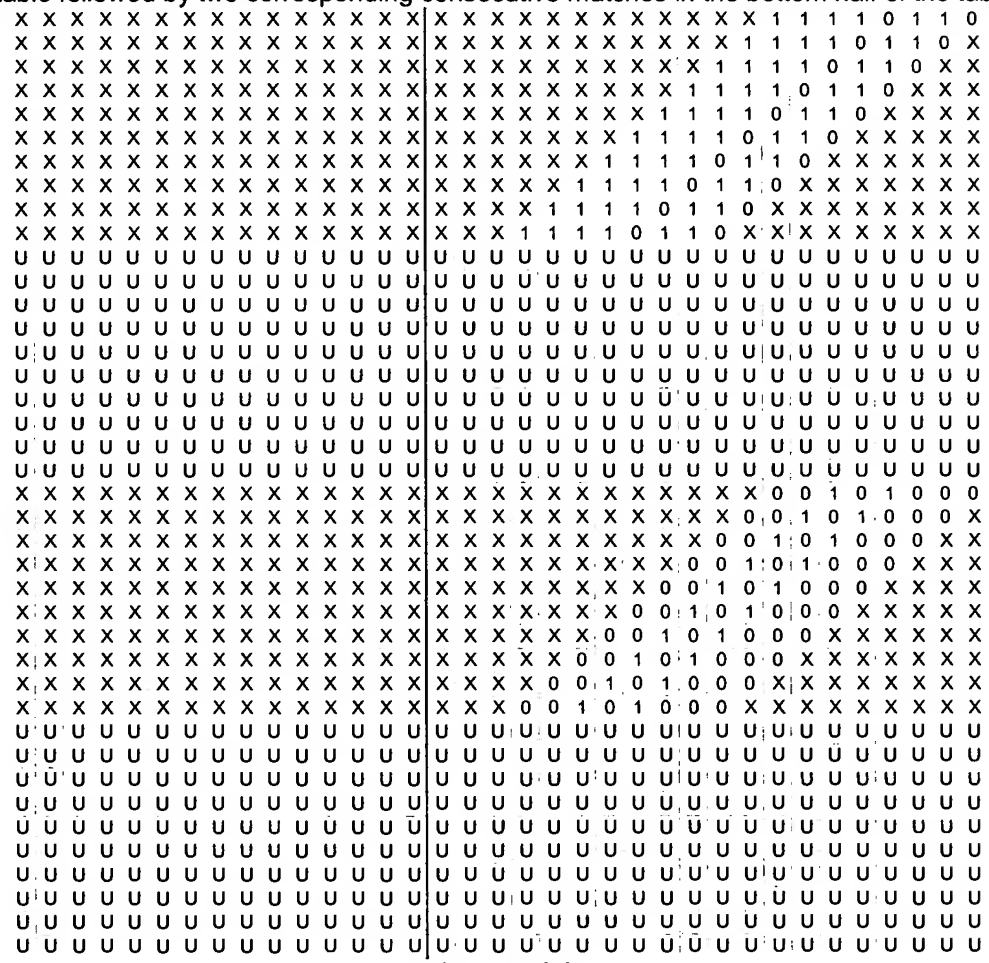

Figure 22

CAM Table for matching multiple F6's followed by multiple 28's for a 8-bit bus width

When matching a SONET framing pattern F6F62828, the state machine would look for two consecutive matches in the top half of the table followed by two corresponding consecutive matches in the bottom half of the table.

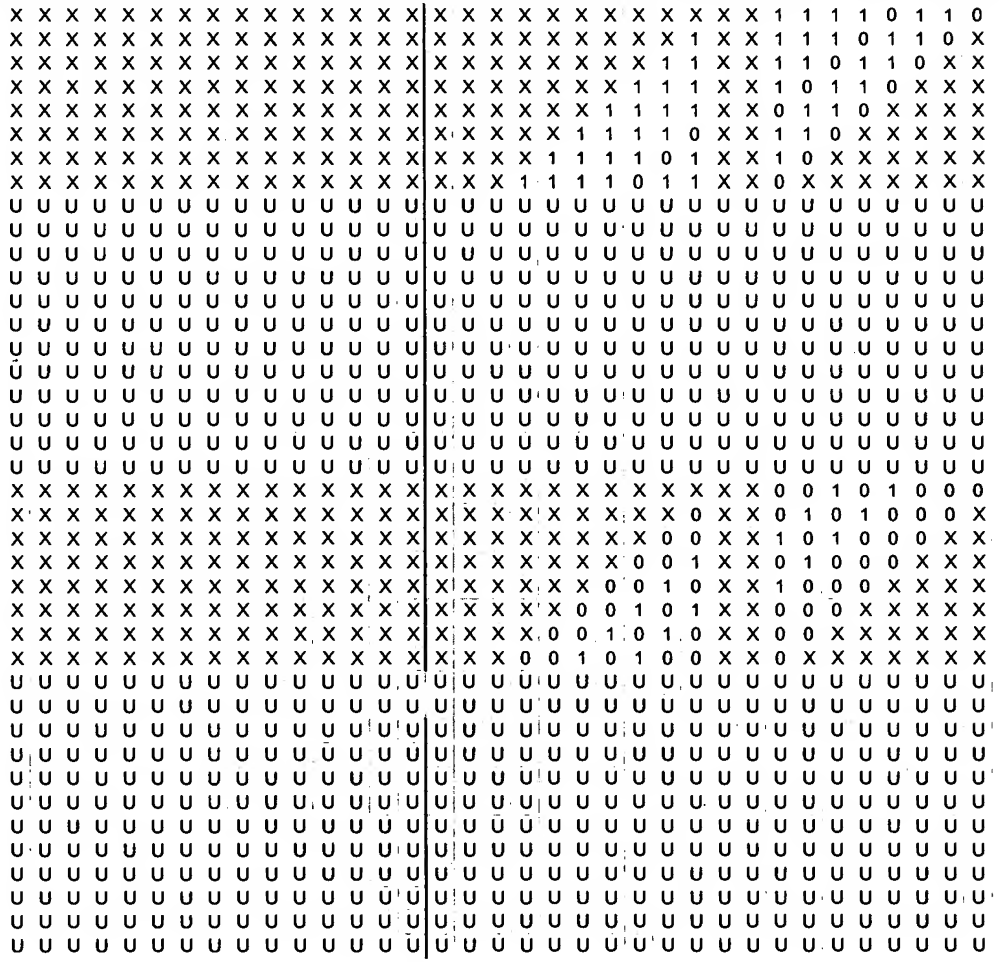

Figure 23

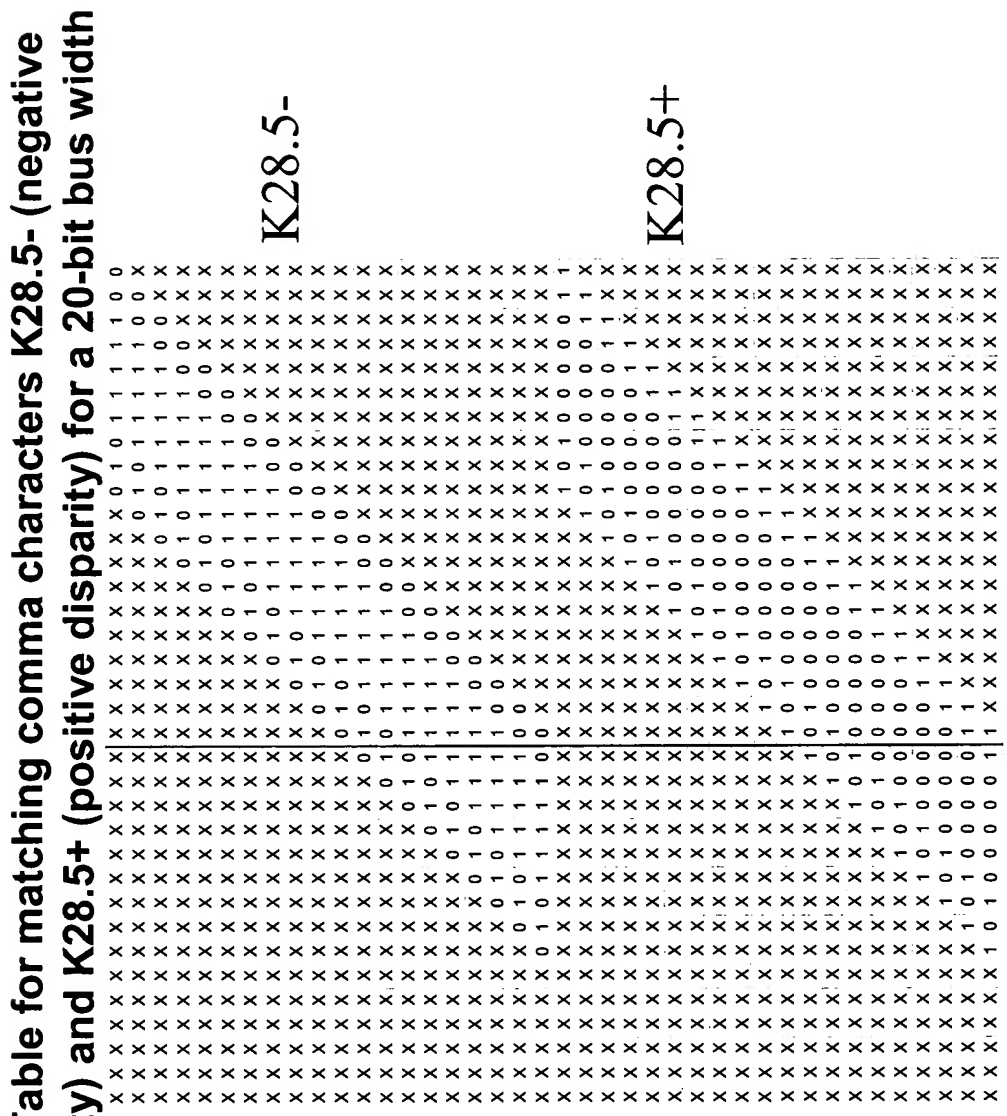
CAM Table for matching comma characters K28.5- (negative disparity) and K28.5+ (positive disparity) for a 20-bit bus width
K28.5-
K28.5+

Figure 24

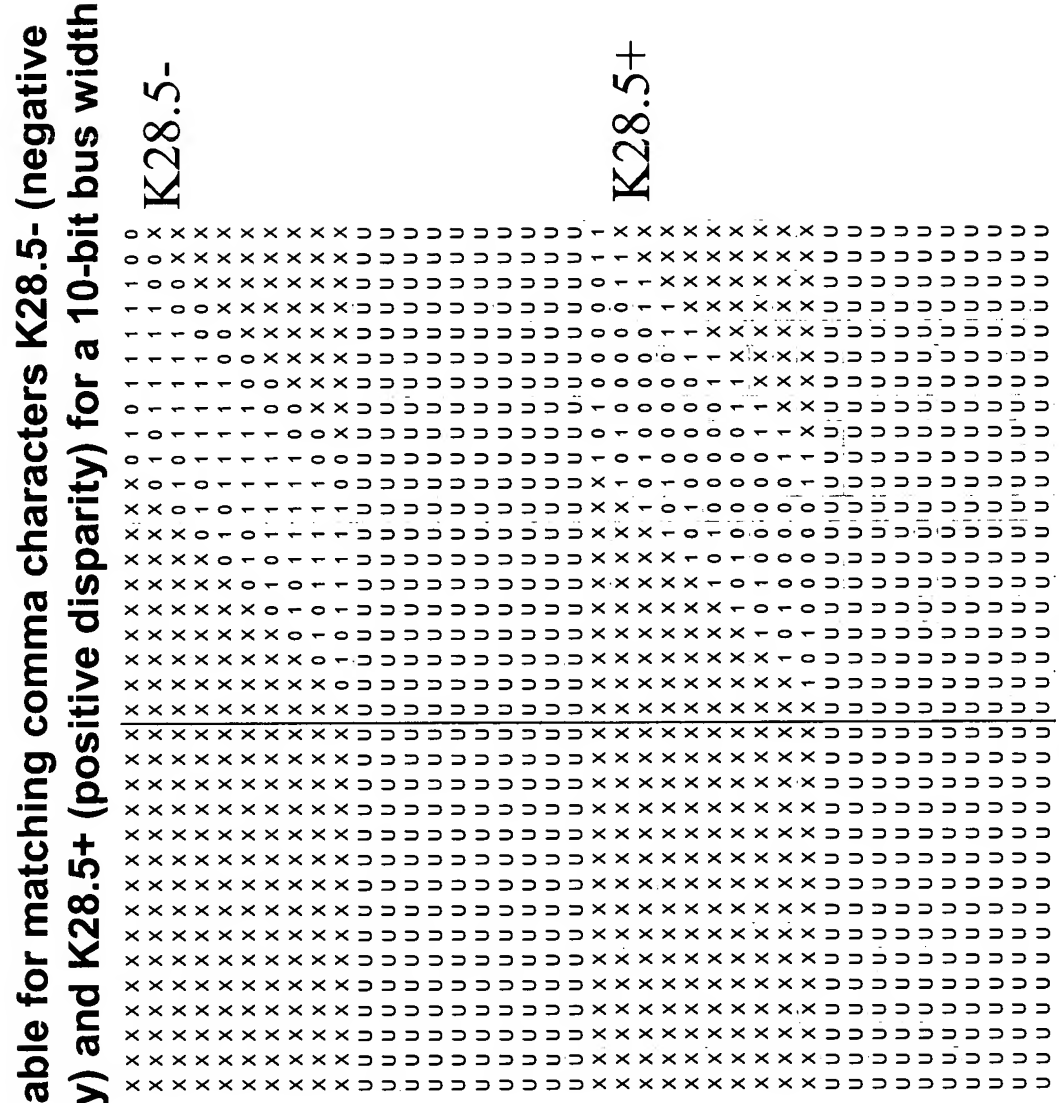
CAM Table for matching comma characters K28.5- (negative disparity) and K28.5+ (positive disparity) for a 10-bit bus width
K28.5-
K28.5+

Figure 25

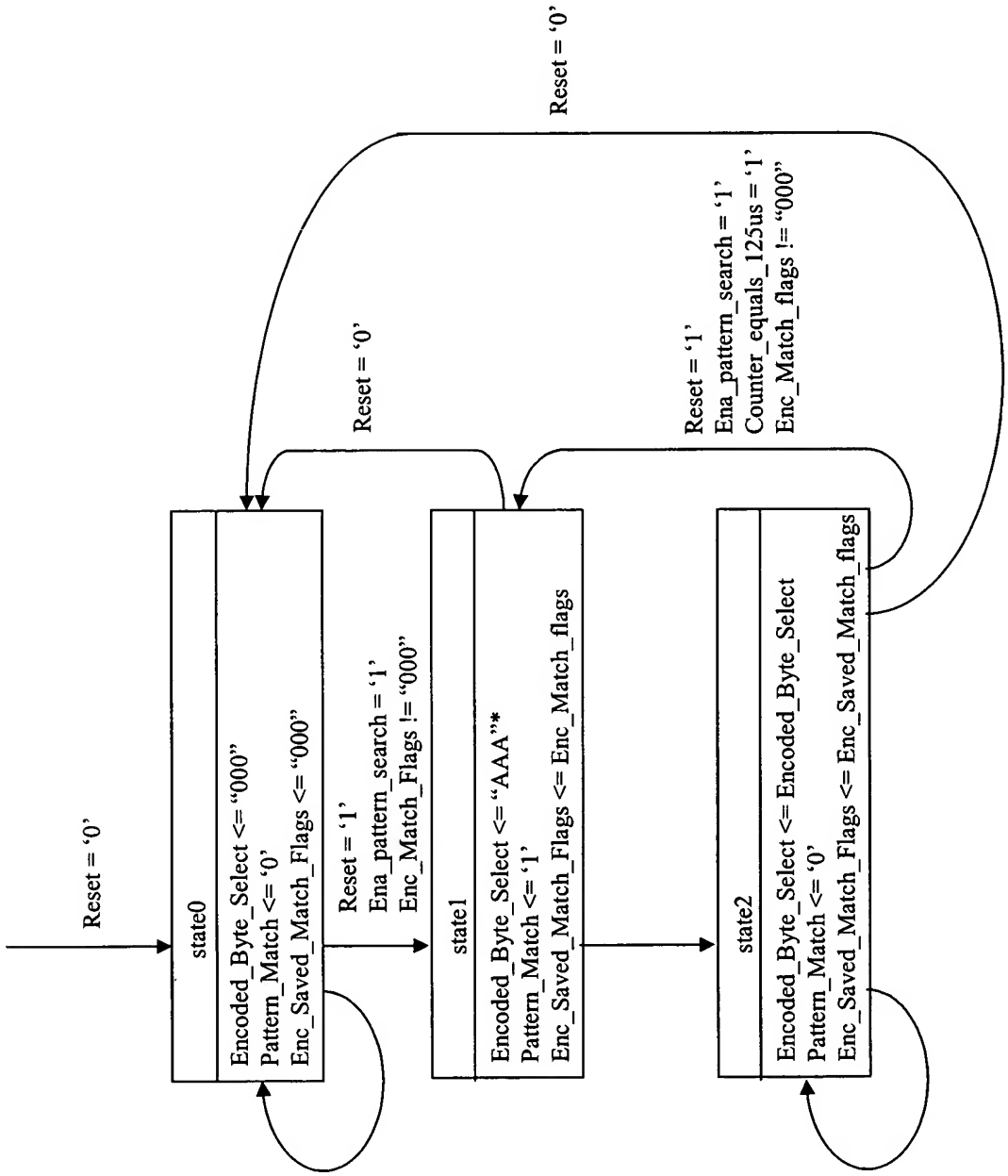

Figure 26

*AAA is the code that selects the right tri_state buffers based upon the CAM Match Flags

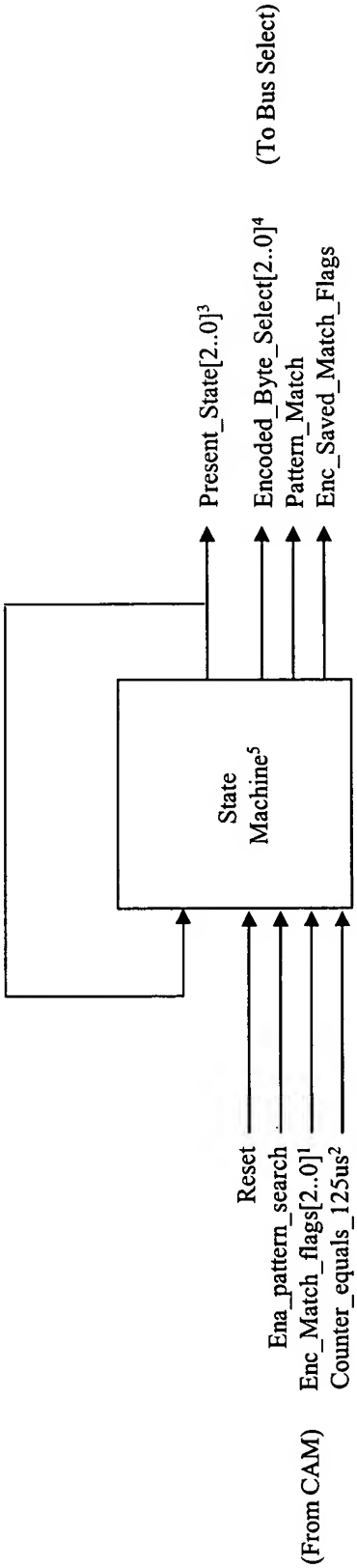

Figure 27

[1]Number of bits will increase to support the binary encoding of the number of CAM table entries.
[2]There is a separate counter that will count to 125 us that is used for framing.
[3]Number of state bits will be determined by the largest number of programmable states that are required to accommodate all desired state machines.
[4]Number of encoded bytes select bits will increase to support a larger bus width.